Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра промышленной электроники

К. В. Бородин

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА И СИСТЕМЫ

Методические указания по выполнению лабораторных работ

Томск 2016

Корректор: А. Н. Миронова

Бородин К. В.

Микропроцессорные устройства и системы : методические указания по выполнению лабораторных работ. – Томск : ФДО, ТУСУР, 2016. – 64 с.

Пособие служит руководством по выполнению лабораторных работ по курсу «Микропроцессорные устройства и системы». Цель работ: усвоение основных принципов работы с программами эмуляции и отладки устройств на микроконтроллере фирмы AVR. Изучаются программная модель, система команд и характеристики периферийных устройств микроконтроллеров ATmega16 с использованием программного продукта Atmel Studio 6.2 для отладки программ на ассемблере и языке Си.

Для студентов вузов радиоэлектронного профиля и инженеровпроектировщиков средств и систем автоматики и промышленной электроники.

> © Бородин К. В., 2016 © ФДО, ТУСУР, 2016

ОГЛАВЛЕНИЕ

1 Программный пакет atmel Studio 4
2 Лабораторный практикум17
2.1 Лабораторная работа № 1 «Порты ввода/вывода»
2.2 Лабораторная работа № 2 «Изучение прерываний, АЦП, UART» 38
2.3 Лабораторная работа № 3 «Таймеры/счетчики,
ШИМ (PWM) модуляция»49
Литература64

1 ПРОГРАММНЫЙ ПАКЕТ ATMEL STUDIO

Atmel Studio – интегрированная среда разработки (IDE) от компании Atmel для разработки приложений под микроконтроллеры ARM Cortex-M и AVR. Ранее назывался AVR STUDIO. В данном пособии приведены экраны программы версии Atmel Studio 6.2. Распространяется свободно с официального сайта компании Atmel.

Программный продукт AVR Studio разрабатывается с 2004 г. Программа позволяет работать как на ассемблере, так и на C/C++, содержит в себе:

- интегрированный компилятор С/С++;
- интегрированный симулятор;
- отладчик (Debugger);
- мастер проектов;
- обширную онлайн-базу готовых примеров;

• поддержка инструментов Atmel, совместимых с 8-разрядной AVR архитектурой, в том числе AVR ONE!, JTAGICE mkI, JTAGICE mkII, AVR Dragon, AVRISP, AVR ISPmkII, AVR Butterfly, STK500 и STK600;

• поддержка плагина AVR RTOS (операционная система реального времени);

• поддержка АТ90РWM1 и ATtiny40;

• программное обеспечение верхнего уровня для поддержки внутрисхемного программирования (In-System Programming, ISP).

Визуальные инструменты позволяют ускорить написание программы. Благодаря связке программных пакетов Atmel Studio и Proteus от фирмы Labcenter Electronics возможно программирование микроконтроллеров без наличия какой-либо материальной базы.

Отладчик AVR Studio имеет два режима работы: режим программной симуляции и режим управления различными типами внутрисхемных эмуляторов (In-Circuit Emulators) производства фирмы Atmel. Важно отметить, что интерфейс пользователя не изменяется в зависимости от выбранного режима отладки.

После запуска AVR Studio для создания нового проекта необходимо в меню **Project** выбрать команду **New Project**. В результате на экране появляется диалоговое окно (рис. 1), в котором необходимо ввести название проекта (**Project name**) и его расположение (**Location**). Новый проект удобнее создавать в отдельной папке.

New Project							? 🔀
Recent Templates		Sort by:	Default		Search Installer	d Templates	δ
Installed Templates C/C++ Assembler Atmel Studio Solution		Sort by: ecc ecc ecc ecc	Default Image: Constraint of the second	C/C++ C/C++ C/C++ C/C++ C/C++	Search Installed Type: C/C+ Creates an AV project	t Templates	RM 32-bit C
					10	#include cavr/i int main(void) Printf("Hello"	o.hs
Name:	GccApplication4						
Location:	C:\AtmelStudioPro	gects_my		Browse			
Solution:	Create new solution	'n		~			
Solution name:	GccApplication4				Create director	y for solution	
						ОК	Cancel

Рис. 1 – Окно создания нового проекта

Далее выбирается тип микроконтроллера. После нажатия кнопки **Finish** на экране появляется окно организации проекта (рис. 2), показывающее все связанные с проектом файлы, и окно для редактирования программы.



Рис. 2 – Окно организации проекта

В это окно для редактирования файла можно с клавиатуры ввести текст программы на языке ассемблера или открыть уже существующий файл (пункт Add Fail в меню **Project**) (рис. 3).



Рис. 3 – Окно редактирования программы на языке С

Все файлы проекта должны быть включены во входной файл проекта с помощью ассемблерной директивы .include. Для смены входного файла проекта на другой надо установить курсор мыши на нужный файл в окне организации проекта и щелкнуть правой кнопкой мыши.

Для осуществления трансляции программы и проверки правильности её написания выбирается пункт **Build** (кнопка **F7**) в меню **Project** (рис. 4).



Рис. 4 – Панель компилятора

Окно View Output содержит сообщения компилятора. В это окно выводится информация о количестве слов кода и данных, о наличии ошибок и другая информация (рис. 5). Выводится информация, сколько процентов использовано в памяти программ (основной код программы) и памяти данных.



Рис. 5 – Окно сообщений компилятора

Для локализации ошибок трансляции в случае их наличия можно в окне сообщений установить курсор мыши на сообщение об ошибке и два раза щелкнуть левой кнопкой мыши. При этом в окне редактирования исходного текста программы курсор будет установлен на строку, вызвавшую сообщение об ошибке, и эта строка будет выделена цветом.

В результате трансляции создается выходной файл в указанном формате.

Для запуска отладчика необходимо выполнить процедуру **Build and Run**, которая вызывается при нажатии на соответствующую кнопку (**F7+Ctrl**) на панели управления. Процедура **Build and Run** выполняется в два этапа. Сперва происходит трансляция входного файла проекта, при которой независимо от установок проекта, кроме выходного файла заданного формата, генерируется и объектный файл. Затем этот объектный файл загружается в отладчик.

Экран AVR Studio в режиме отладки представлен на рис. 6.

8



Рис. 6 – Экран Atmel Studio в режиме отладки

При выполнении процедуры **Build and run** (или при загрузке объектного файла) автоматически открывается окно исходного текста исполняемой микроконтроллером программы.

После выполнения процедуры появляется желтая стрелка, указывающая позицию программного счетчика микроконтроллера (рис. 7). Этот указатель всегда находится на строке, которая будет выполнена в следующем цикле.

Пользователь может выполнять программу полностью в пошаговом режиме, трассируя блоки функций или выполняя программу до того места, где стоит курсор. В дополнение можно определять неограниченное число точек останова, каждая из которых может быть включена или выключена. Точки останова сохраняются между сессиями работы.

В Atmel Studio для отладки программы предусмотрены две команды пошагового режима: **Step Over** и **Step Into**. Разница между ними в том, что команда Step Over не работает в подпрограммах. С помощью команд

пошагового режима можно проследить изменения значений в регистрах устройств ввода/вывода, памяти и регистрового файла. К командам шагового режима относятся также **Auto Step** и **Multi Step**. Помимо шагового режима возможна отладка программы с использованием точек останова (**Breakpoints**). Командой **Go** запускается исполнение программы. Программа будет выполняться до остановки пользователем или до обнаружения точки останова.



Рис. 7 – Окно исходного текста программы в режиме отладки

Для установки точки останова в Atmel Studio служит пункт меню **Debug -> Toggle Breakpoint**. Точка останова ставится в строке, отмеченной курсором. Красная отметка в левом поле окна исходного текста программы показывает установленную точку останова.

Имеется возможность отлаживать транслированный ассемблерный код программы. Для этого в окне текста нужно выбрать вкладку Disassembly (рис. 8).



Рис. 8 – Окно дизассемблера с программным кодом

В процессе отладки также можно выбрать пункт меню **Debug** -> **Run To Cursor (Ctrl+F10)**. При выборе этого пункта исполняемый код выполняется до достижения команды, обозначенной курсором. При этом если отладчик обнаруживает точку останова, установленную ранее положения курсора, то останов будет выполнен только в случае его разрешения в окне **Debug Option**, в противном случае выполнение не приостанавливается. Если команда, обозначенная курсором, не достигается, отладчик продолжает исполнять код программы до тех пор, пока исполнение не будет

прервано пользователем. Поскольку режим **Run To Cursor** зависит от позиции курсора, он доступен только при активном окне исходного текста.

Для остановки исполнения программы пользователем служит команда **Break (Ctrl+F5)**. В состоянии останова эта команда недоступна. При отладке с использованием точек останова, или если адрес останова указан курсором в окне исходного текста, модификация информации во всех окнах происходит только при достижении останова (или при прекращении исполнения программы пользователем).

Пункт меню **Debug** -> **Reset** (Shift+F5) выполняет сброс микроконтроллера. Если программа при этом выполняется, то ее исполнение будет остановлено. После сброса информация во всех окнах модифицируется.

Для наблюдения за работой программы можно открыть несколько окон, отображающих состояние различных узлов микроконтроллера. Окна открываются нажатием соответствующих кнопок на панели инструментов или при выборе соответствующего пункта меню **View**.

Регистровый файл микроконтроллера AVR отображается в окне Work space (вкладка I/O, рис. 9), а также можно открыть отдельное окно Registers (рис. 9) Если в процессе выполнения программы в очередном цикле значение какого-либо регистра изменится, то этот регистр будет выделен красным цветом. При этом если в следующем цикле значение регистра останется прежним, то цветовое выделение будет снято. Такое же цветовое выделение реализовано в окнах устройств ввода/вывода, памяти и переменных.

Также в окне I/O WorkSpace отображается состояние встроенных периферийных устройств микроконтроллера (рис. 10).



Рис. 9 – Окно состояния регистрового файла и окно I/O WorkSpace

IO View						Ψ×
Filter:			•	<u></u>		
Nan	ne		Valu	e		
EEPROM						~
🗉 🛃 EXTERNAL_I	NTERRUPT					
JTAG						
VO PORTA						
PORTB						
PORTC						
PORTD						
🕀 🗎 SPI						
∃ O TIMER_COUNT	VTER_0					E
∃	NTER_1					
∃ U TIMER_COU	NTER_2					
TWI						
🗉 🗎 USART						
E 🗎 WATCHDOG						×
Name	Address	Value		Bits		
110 PINB	0x36	0x00				^
10 DDRB	0x37	0x00				
VO PORTB	0x38	0xFF	000			
		_	_			$\mathbf{\Sigma}$
🔄 IO View 🔍 AS	FEx 🕴	Proces	sor 🏹 :	Solution	Prop 😭	erti

Рис. 10 – Развернутый порт РОКТВ и отображение его регистров PINB, DDRB, PORTB

В этом окне отражаются все функциональные блоки *выбранного* микроконтроллера. Если выбрать другой контроллер, то часть функционала может отсутствовать. Любой блок может быть раскрыт нажатием на его значок. При раскрытии блока в окне отражаются адреса и состояния всех его регистров и отдельных, доступных для модификации, битов (рис. 11). Каждый доступный для модификации бит может быть установлен или сброшен как программой по ходу ее исполнения, так и пользователем вручную (указав курсором нужный бит и щелкнув левой кнопкой мыши, пользователь может изменить значение бита на обратное) – в режиме программной симуляции это является способом имитации входного воздействия на микроконтроллер. Например, имитируется нажатие кнопки или пришедшие данные в буфер UART, АЦП и др.

IO View				• 4 ×								
Filter:			- 🥒									
Na	ame		Value									
BOOT_LOA	D			<u>^</u>								
🖃 🛄 CPU												
Sleep N	Iode Select	(MCUC	0x00 🔽									
🛃 Interrupt Sense Control 1 🛛 0x00 💌												
Interrupt Sense Control 0 0x00 V												
Name	Address	Value	Bits									
🕀 🗎 OSCCAL	0x51	0x00	00000000	^								
🗉 🗎 MCUCSR	0x54	0x01										
🗉 🗎 MCUCR	0x55	0x00	000000000									
SP	0x5D	0x045	00000000 00000000	_								
🖃 🗎 SREG	0x5F	0x80										
I		0x01										
E T		0x00		=								
ШH		0x00										
i≣ s		0x00										
l v ⊡		0x00										
III N		0x00										
III Z		0x00										
E C		0000		<u>×</u>								
🔄 IO View 🔍 A	SFEx I	Proces	ssor 🔍 Solution 🖀 Pi	roperti								

Рис. 11 – Окно состояния процессорного ядра

Для индикации состояния программного счетчика, указателя стека, содержимого регистра статуса SREG и индексных регистров X, Y и Z в процессе отладки программы предназначена вкладка **Processor** в окне I/O WorkSpace.

В этом же окне отображается текущее время выполнения программы и тактовая частота ядра микроконтроллера.

Просмотр ячеек памяти программ, памяти данных, EEPROM и регистров портов ввода/вывода в ходе исполнения программы осуществляется также с помощью диалогового окна **Memory**. Выпадающее меню диалогового окна позволяет выбрать один из четырех массивов ячеек памяти: Data, IO, Eeprom, Program Memory. Для одновременного просмотра нескольких областей окно **Memory** может быть открыто несколько раз. Информация в диалоговом окне может быть представлена в виде байтов или в виде слов в шестнадцатеричной системе счисления, а также в виде ASCII-символов (рис. 12).

Memor	Memory 1 👻 🕂 🗙													
Memo	ry:	prog i	FLAS	н						Ŧ			••	
data	0x0	000	00	00	00	00	00	00	00	00	00		~	
data	0x0	009	00	00	00	00	00	00	00	00	00			
data	0x0	012	00	00	00	00	00	00	02	01	00			
data	0x0	01B	00	5f	04	00	00	00	f8	fe	ff	шюя		
data	0x0	024	00	00	00	00	00	00	00	20	00			
data	0x0	02D	00	00	00	02	ff	02	00	00	00	я		
data	0x0	036	00	00	ff	00	00	00	00	00	00	я	_	
data	0x0	03F	00	00	00	00	00	00	00	00	00			
data	0x0	048	00	00	00	00	00	00	00	00	00			
data	0x0	051	00	00	00	01	00	00	00	20	00			
data	0x0	05A	00	00	00	5d	04	80	00	00	00].ъ		
data	0x0	063	00	00	00	00	00	00	00	00	00			
data	0x0	06C	00	00	<u>00</u>	00	00	00	00	00	00		\sim	
🔲 м	1	🕺 R		в		I M	1	a (.	2	C	/ 🖅 I 🖉 🖉	D	

Рис. 12 – Окно просмотра содержимого памяти

Чтобы внести изменения в ячейке памяти, достаточно дважды щелкнуть мышкой по данной ячейке.

Для внесения изменений в программу в процессе отладки необходимо редактировать её исходный текст. При попытке запуска симулятора на исполнение программы после редактирования на экране появляется окно, сообщающее об изменении программы и необходимости её компиляции.

Для сохранения проекта необходимо воспользоваться пунктом меню **Project -> Close**. При закрытии проекта сохраняются все его настройки. Во время следующей загрузки настройки будут автоматически восстановлены.

2 ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Выбор варианта лабораторных работ осуществляется по общим правилам с использованием следующей формулы:

$$V = (N \times K) \text{ div } 100,$$

где *V*-искомый номер варианта,

N – общее количество вариантов,

div – целочисленное деление,

при V = 0 выбирается максимальный вариант,

К-код варианта.

2.1 Лабораторная работа № 1 «Порты ввода/вывода»

Цель работы

Целью лабораторной работы является изучение простейших команд языка С, портов ввода/вывода и отладка прикладных программ для микроконтроллера AVR семейства MEGA с помощью персонального компьютера и программного пакета Atmel Studio 6.2 (не ниже).

Краткая теория

Порты ввода/вывода AVR имеют число независимых линий «вход/выход» от 3 до 53. Каждая линия порта может быть запрограммирована на вход или на выход. Мощные выходные драйверы обеспечивают токовую нагрузочную способность 20 мА на линию порта (втекающий ток) при максимальном значении 40 мА, что позволяет, например, непосредственно подключать к микроконтроллеру светодиоды и биполярные транзисторы.

Общая токовая нагрузка на все линии одного порта не должна превышать 80–200 мА (в зависимости от корпуса и порта), а на все линии всех портов 400 мА (все значения приведены для напряжения питания 5 В). Каждый порт обслуживают три регистра: регистр данных, регистр направления и регистр выводов.

Управление, настройка, контроль и другие всевозможные операции с выводами AVR-микроконтроллеров осуществляются всего лишь через три регистра:

- DDRx (направления);

- PORTx (настройки);

- PINx (состояния).

В работе Вам потребуется установить или сбросить бит/биты в разных регистрах. Сделать это можно с помощью логических операций «ИЛИ» и «И», используя их свойства – любое значение «ИЛИ» 1 равно 1, любое значение «И» 0 равно 0.

Например, для установки 3-го бита в регистре PORTA можно написать:

PORTA |= (1 << 3);

Данная запись означает для компилятора следующее:

1) константу – 0x01 (0b0000001) сдвинуть на 3 разряда влево. Получается 0x08 (0b00001000);

2) сделать операцию «ИЛИ» между значением регистра PORTA и результатом предыдущей операции;

3) записать результат предыдущей операции обратно в регистр PORTA.

Таким образом, если нам необходимо установить несколько бит в регистре, можно написать следующим образом:

PORTA $|= (1 \le 3)|(1 \le 4)|...|(1 \le 1);$

Для *сброса* бита в регистре необходимо воспользоваться операцией «И» со значением, содержащим нули в интересующих нас позициях. Этого можно добиться с помощью операции инвертирования – «~».

PORTA &= ~(1 << 3); PORTA &= ~((1 << 3)|(1 << 4)|...|(1 << 1)); Светодиодный семисегментный индикатор представляет собой группу светодиодов, расположенных в определенном порядке и объединенных конструктивно (рис. 13). Зажигая одновременно несколько светодиодов, можно формировать на индикаторе цифры либо некоторые буквы. Индикаторы различаются по типу соединения светодиодов – общий анод, общий катод; по количеству отображаемых разрядов – однораразрядные, двухразрядные и т. д.; по цвету – красные, зеленые, желтые и т. д.



Рис. 13 - Обозначение и внешний вид семисегментных индикаторов

Чтобы зажечь на индикаторе какую-то цифру, нужно настроить порты, к которым подключен индикатор, на выход и установить в порту микроконтроллера код нужной цифры.

Пусть семисегментный индикатор подключен напрямую к порту (вывод А индикатора = младшему биту порта (0), вывод DP = старшему (7)). Тогда массив кодов цифр будет представлять:

	unsigned char number[] =
{	
	0x3f, //0
	0x06, //1
	0x5b, //2
	0x4f, //3
	0x66, //4
	0x6d, //5
	0x7d, //6
	0x07, //7
	0x7f, //8
	0x6f //9
};	

Используя десятичные цифры от 0 до 9 в качестве индекса, можно выводить в порт нужные коды командой.

PORTB = number[1]; //вывод числа 0x06 в порт, которое соответствует цифре 1.

Программа работы

Запустить программу Atmel Studio 6.2 (не ниже), ярлык которой расположен в меню Atmel -> Atmel Studio (рис. 14). В работе представлены экраны программы Atmel Studio 6.2.



Рис. 14 – Меню запуска программы Atmel Studio 6.2

После запуска откроется основное стартовое окно компилятора (рис. 15). Язык программы английский, как и других программ для программирования микроконтроллеров. На стартовой странице можно видеть ранее созданные проекты, часто задаваемые вопросы и примеры, в том числе видеопримеры. Знакомство с незнакомым программным продуктом лучше всего начать с примеров.



Рис. 15 - Стартовое окно программы

Выбрать **New Example project**. Откроется окно выбора примеров проектов (рис. 16). Данная база загружается с сайта ATMEL (требует интернет-соединения) и постоянно пополняется новыми примерами. В ней представлены различные отлаженные примеры отдельных блоков и рабочих больших комплексных проектов.

В верхней части окна выбрать готовый пример для ATmega16:

megaAVR, 8-bit, Applications, megaAVR GPIO example – STK600 – ATMEGA16.

В нижней части указать имя проекта и путь. Они не должны содержать русских символов. Нажать **ОК** при готовности.

New Example	Project from ASF or Extensions	
Device Family: m All Projects Kit Category Technology	Attract of Extensions egaAVR, 8-bit Category: Applications Search for Example Projects All Drivers Components MEGA-1284F Services MEGA-1284F Services n - MEGA-1284P Xplained megaAVR GFIO example - STK600 - ATmega1284P megaAVR GPIO example - STK600 - ATmega166 megaAVR GPIO example - STK600 - ATmega168A megaAVR GPIO example - STK600 - ATmega169P megaAVR GPIO example - STK600 - ATmega2560 megaAVR GPIO example - STK600 - ATmega324A megaAVR GPIO example - STK600 - ATmega328 megaAVR GPIO example - STK600 - ATmega48A megaAVR GPIO example - STK600 - ATme	 megaAVR GPIO example - STK600 - ATmega16 This application show how to set use the different GPIO features on megaAVR. [megaAVR GPIO example - STK600 - ATmega16] Online Help Online Help STK600
Project Name:	MEGA_GPIO_EXAMPLE4	
Location:	C:\AtmelStudioProgects_my	Browse
Solution:	Create New Solution	
Solution name:	MEGA_GPIO_EXAMPLE4	
Device:	ATmega16	
		OK Cancel

Рис. 16 – Окно выбора контроллера

Откроется выбранный пример использования портов GPIO (рис. 17).



Рис. 17 – Основное рабочее окно программы

В выбранном примере показана обработка внешнего прерывания INTO при нажатии кнопки. Работа с прерываниями рассматривается во второй и третьей лабораторных работах.

Задание 1. Знакомство с компилятором и режимом отладки (об-

щее для всех вариантов)

Найти и удалить из проекта следующий код:

```
// Toggle both pin PD0 and PD1, which toggles LED0 and LED1.
#if defined(__AVR_ATmega16__) || defined(__AVR_ATmega32__)
            || defined( AVR ATmega64 ) || de-
fined(__AVR_ATmega128__)
/* For older megaAVR devices read-modify-write PORT register.
      * This isn't safe for interrupts.
      */
      PORTD ^= (1 << PIND0) | (1 << PIND1);
#else
      // Use PIN register to toggle on newer megaAVR devices
      PIND = (1 << PIND0) | (1 << PIND1);
#endif
// Only use Pin Change Interrupt handler for devices supporting this.
#ifdef EXAMPLE PCICR
/* Enable pin change interrupt for PB0 which is controlled by SW0
* First we need to enable pin change interrupt for the wanted port.
      */
      EXAMPLE PCICR = (1 << EXAMPLE PCIE);
// Then we need to set the pin change port mask to get the bit we want.
      EXAMPLE_PCMSK = (1 << PCINT0);
#endif
      // Enable interrupts
      sei();
/* Busy loop, and a breakpoint can be used in the interrupt handler to
      * see interrupts being triggered by SW0.
      */
      while (true);
```

Заменить

```
PORTD &= ~(1 << PORTD1);
```

на

```
PORTD &= ~(1 << PORTD0);
```

Добавить в бесконечный цикл while(1)



Проект готов для отладки и симуляции.

Нажать на кнопку **Build Solution (F7)**, что на панели компиляции (рис. 18), и **Старт (Start Debugging F5)** (рис. 19).



Рис. 18 – Панель компиляции



Рис. 19 – Панель отладки («дебага»)

Появится предупреждение компилятора (рис. 20). В нем указано, что не найден подключенный по USB программатор и не определена плата для программирования. Указать Simulator в качестве программатора, таким образом, на компьютере будет происходить имитация подключенной отладочной платы (рис. 21).

MEGA_GPIO_EXAM	PLE4* X mega_gpio_example.c ASF	F Wizard
Build		Platform: N/A
Build Events		
Toolchain	Selected debugger/programmer	
Device	×	Atmel Studio
Tool		Please select a connected tool and interface and try again.
Advanced	Programming settings	
	Erase entire chip	Continue

Рис. 20 – Окно выбора программатора

MEGA_GPIO_EXAM	PLE4* X mega_gpio_example.c
Build	Configuration: N/A
Build Events	
Toolchain	Selected debugger/programmer
Device	Simulator 🗸
Tool	Simulator
Advanced	Programming settings
	Erase entire chip 🖌
	Preserve EEPROM
	Select Stimuli File for Simulator
	Stimuli File Activate stimuli when in breakm

Рис. 21 – Выбор симулятора

Необходимо открыть окно периферии микроконтроллера. Для этого зайти в меню **Debug -> Windows -> I/O View** и выбрать место размещения. Обычно оно располагается в правой части экрана (рис. 22).

MEGA_GPIO_EXAMPLE4 (Debugging) - At	mel	itudio				
File Edit View VAssistX ASF Project Build	Deb	ug Tools Window Help				
		Windows	•	3	Breakpoints	Alt+F9
i 🔁 📴 🐺 🍋 🔓 😭 💁 💒 🗐 💷 🗉	ÞII	Start Debugging and Break	Alt+F5		Data Breakpoints	
		Stop Debugging	Ctrl+Shift+F5		Processor View	
	$= \triangleright$	Start Without Debugging	Ctrl+Alt+F5		I/O View	
P main.while P while * But a typical pequiperent		Disable debugWIRE and Close			Live Watch	
* don't require these, but		Continue	F5		Output	
* on the safe side.	Ę	Execute Stimulifile		3	Parallel Tasks	Ctrl+Shift+D, K
PORTB = Øxff;	Ę	Set Stimulifile		ふ	Parallel Stacks	Ctrl+Shift+D, S
	5	Restart			Watch	•
/* Read STK600 switches on p *	11	Break All	Ctrl+F5		Autos	Ctrl+Alt+V, A
* Press any of the STK600 s	61	QuickWatch	Shift+F9	-	Locals	Alt+4
* A press will be seen as b	<u>چ</u>	Step Into	F11		Immediate	Ctrl+Alt+I
*/	Ç⊒	Step Over	F10	ça,	Call Stack	Alt+7
val = PINB;	2	Step Out	Shift+F11	-	Threads	Ctrl+Alt+H
while(1)	*1	Run To Cursor	Ctrl+F10	***	Modules	Ctrl+Alt+U
{	Î	Reset	Shift+F5		Processes	Ctrl+Shift+Alt+P
/* It's also possible to *		Percepio Trace	•		Memory	•
* Setting pin PD0 high		Toggle Breakpoint	F9	Z	Disassembly	Alt+8
*/		New Breakpoint	•	öx	Registers	Alt+5
PORTD = (1 << PORTD0);	ò	Delete All Breakpoints	Ctrl+Shift+F9			
// Setting pin PD1 low i	0	Disable All Breakpoints			=	
PORTD &= ~(1 << PORTD1);		Clear All DataTips				

Рис. 22 – Окно выбора периферийных устройств

Выставить точки остановки («брейкпоинты») и открыть регистры состояния порта D (окно I/O View), как указано на (рис. 23). Для того чтобы установить точки останова, нужно в левой части экрана в серой области нажать мышкой. Красный круг обозначает место остановки программы, стрелочка указывает следующую команду микроконтроллера.



Рис. 23 – Программа с установленным брейкпоинтом

Нажать **F5** для перехода от одной точки к другой. Понаблюдать и дать комментарий, как выполняется программа работа с битами (рис. 24). Красным цветом обозначено измененное значение на порту (ножке микроконтроллера).



Рис. 24 – Выставление бита

Перейти на страницу дизассемблера, нажав на вкладку **Disassemble** в верхнем левом углу (рис. 25).

\$	MEGA_GPI	0_EX	AMPLE3 (Debu	gging) - AtmelStudi	o		
File	e Edit Viev	N VA	sistX ASF Pro	ject Build Debug 1	Fools Wind	ow	Help
	ヨー田 治	1 - 12		a 🙉 🔊 - (° - ,	g - e	8	a, a) Mu Debua 🚽 🦓 🗒 🗄 🚈 🚈 🖄 🗆 💭 🙄
		. <u> </u>			N	<u>-</u>	
	🖌 🖾 😵 '	*** 6*	🛅 🦄 💑 🖽		P 63	23	: Ç≓ °≣ *≣ 1: Hex 🔍 * ╤ : 🔃 📖 🛄 🛄 🛄 ╤ : 🔤 ╤ : ;
Dis	sassembly >	meg	a_gpio_example.c	conf_example.h		•	IO View 👻 🕂 🗙
A	ddress: mair	n				•	Filter:
$\overline{\mathbf{v}}$	Viewing Op	tions					Name Value
	00000039	POP	RØ Po	p register from s	tack	~	🗉 🗄 AD_CONVERTER 🔼 🔼
	0000003A	POP	R1 Po	p register from s	stack	-	■ COMPARATOR
	0000003B	RET	Interr	upt return			🕀 🛄 CPU
	DDRD	= 0x1	f;				EEPROM
	0000003C	SER	R24 Se	t Register			EXTERNAL_INTERRUPT
	0000003D	OUT	0x0A,R24	Out to I/O loca	ation		VO PORTB
	PORTE) = 0)	df;				PORTC I
	0000003E	OUT	0x0B,R24	Out to I/O loca	ation		VO PORTD
	PORTE) = 0					🗉 🗎 SPI
	0000003F	OUT	0x0B,R1	Out to I/O loca	ation		
	PORTE	3 = 0	df;				
	00000040	OUT	0x05,R24	Out to I/O loca	ation		TIMER COUNTER 2
	val =	= PINE	3;				
	00000041	IN F	24,0x03	In from I/O loc	ation	-	
	C:\At	tmelSt	udioProgects	_my\MEGA_GPIO_EXA	MPLE3\ME	_	
	F	PORTD	= (1 << POR	RTD0);			Name Address Value Bits
	00000042	SBI	0x0B,0	Set bit in I/O	register		PIND 0x29 0x01
Ι.	F	PORTD	&= ~(1 << PO	ORTD0);			DDRD 0x2A 0xFF
->	00000043	CBI	0x0B,0	Clear bit in I/	0 regist		PORTD 0x2B 0x01
	00000044	RJM	PC-0x0002	Relative ju	ımp		
	No so	ource	file				×
	00000045	CLI	Global	Interrupt Disabl	le		🔄 IO View 🔍 ASF Ex 🐺 Processor 🛛 Solution 🖀 Properti
	00000046	RJM	PC-0x0000	Relative ju	ımp		
	00000047	NOP	Undefi	ined			Registers $\checkmark \psi \times$
	00000040	NOP	Undefi	ned			R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00
	00000049	NOP	Undefi	ned			R04 = 0X00 R05 = 0X00 R06 = 0X00 R07 = 0X00
	0000004A	NOP	Undefi	ned			R08 = 0.000 R09 = 0.000 R10 = 0.000 R11 = 0.000
	00000040	NOP	Undefi	ned			R12 = 0000 R13 = 0000 R14 = 0000 R15 = 0000
	0000004C	NOP	Undefi	ned			RI0 = 0.000 RI7 = 0.001 RI0 = 0.000 RI9 = 0.000
	00000045	NOP	Undefi	ned			$R_{20} = 0.000 R_{21} = 0.000 R_{22} = 0.000 R_{23} = 0.000$
	0000004F	NOP	Undefi	ned			$P_{24} = 0.000 R_{23} = 0.000 R_{20} = 0.000 R_{27} = 0.001$
1							R31 = 0x00
-					·		
Wa	atch 2				- ₽	x	
	Auton 🔜	Locala	E Match 1	Watch 2			🔤 Re 💭 Br 🗮 Me 🌾 Ca 🎦 Co 🍘 Im 🗐 Ou
Sto	pped						

Рис. 25 – Код ассемблера

Дать комментарий, какие команды появились вместо команд:

PORTD |= (1 << **PORTD**0);

PORTD &= ~(1 << PORTD0);

Найти их в списке команд ассемблера. Дать комментарий, что выполняют команды NOP в конце вашей программы Задание 2. Знакомство с регистрами порта (общее для всех вариантов)

Модифицировать программу в цикле **while(1)** так, чтобы при пошаговом режиме выполнения инвертировались значения всех светодиодов в порту **PORTD** (рис. 26). Привести код ассемблера.

Na	ame	Address	Value	Bits				Name	Address	Value	Bits	
1/C 1/C	PIND DDRD PORTD	0x30 0x31 0x32	0xFF 0xFF 0xFF			<		VO PIND VO DDRD VO PORTD	0x30 0x31 0x32	0x00 0xFF 0x00		
I 0	Vi 🔍	ASF	Pro	ce 💐 Solut	i 🚰 Pro		<->	IO Vi	ASF	Pro	oce 🟹 Soluti 💣 F	Prop

Рис. 26 – Изменение состояния на портах в задании 2

Задание 3. Создание переменных, изучение условий if(), циклов

Объявить переменную і в процедуре main,

```
int main(void)
{
    // Variable to put switch input into
    uint8_t val;
    uint8_t i;
    //
```

Модифицировать код в бесконечном цикле while(1)

```
while(1)
{
    PORTD |= (1 << i);
    i++;
}
```

Установить брейкпоинт на добавленной команде (рис. 27).



Рис. 27 – Основной бесконечный цикл программы

Нажимая кнопку **F10** (пошагово) пронаблюдать новое поведение на порте (рис. 28)

Na	me	Address	Value	Bits	
1/0	PIND	0x30	0x1F		
1/0	DDRD	0x31	0xFF	00000000	
1/0	PORTD	0x32	0x1F		
IO V	/i 🔍	ASF	Proc	ce 💐 Soluti 督 I	Prop

Рис. 28 – Регистры порта D

Добавить условие if(), понаблюдать за поведением на порте D.

Посмотреть код ассемблера. Найти в таблице команд ассемблера, какой код появился в месте условий if().

```
while(1)
{
    PORTD |= (1 << i);
    i++;
    if (i>8)
    {
        i=0;
        PORTD =0;
    }
}
```

Выполнить задание согласно варианту:

1. Реализовать бегущий огонек, который доходит до 8-го разряда и начинает с 1-го (...0100 0000 -> 1000 0000 -> 0000 0001...).

2. Реализовать бегущий огонек, который доходит до 8-го разряда и начинает считать в обратную сторону (...0100 0000 -> 1000 0000 -> 0100 0000 -> 0010 0000...).

4. Реализовать эффект ёлочки, который доходит до 8-го разряда и начинает считать с начала (...0111 1111 -> 1111 1111 -> 0000 0000 -> 0000 0001...).

6. Реализовать инверсный эффект эквалайзера (см. п. 5, заменив 0 на 1).

7. Бегущий огонек на линиях порта С должен сменить направление, если на пяти линиях порта В установлены логические единицы.

8. Ёлочка 2 (линейка светодиодов, подключенная к линиям порта, последовательно заполняется огнями и постоянно горит, а звезда – старший бит – моргает).

9. «Бегущий огонёк» в одну сторону бежит по выводам порта В, в другую – по выводам порта D.

10. Если поступившее на выводы порта D – число чётное, то должны «мигать» выводы порта B, если нет – то порта C.

Задание 4. Изучение кнопки

Модифицировать код программы в бесконечном цикле.

```
while (1)
{
    if(PIND & (1<<PIND3))// если флаг в регистре RegX установлен
    {
        PORTD |= (1 << PORTD0);
    }
    else
    PORTD &= ~(1 << PORTD0);
}</pre>
```

Установить брейкпоинт и зайти в режим отладки (рис. 29). Нажимая кнопку **F10** (пошагово), пронаблюдать новое поведение на порте.



Рис. 29 – Основной цикл программы

Нажать на 3-й пин порта D, имитировав нажатие кнопки (рис. 30).

Name	Address	Value	Bits	
1/0 PIND	0x29	0x00	00000000	^
10 DDRD	0x2A	0xFF	0000 <u>0</u> 000	
VO PORTD	0x2B	0x08		
			Bit 3	\sim
🖪 IO View 🔍 A	SF Ex 🖡	Proces	ssor 💐 Solution 督 Proper	ti

Рис. 30 – Имитация нажатия кнопки

Нажимая кнопку **F10** (пошагово) пронаблюдать новое поведение на порте (рис. 31).



Рис. 31 – Отладка программы

Был выставлен бит на 0-й ножке.

Снова нажать на PIN 3, имитировав отпускание кнопки (рис. 32).

Name	Address	Value	Bits
10 PIND	0x29	0x09	
1/0 DDRD	0x2A	0xFF	
VO PORTD	0x2B	0x01	

Рис. 32 – Имитация отпускания кнопки

Нажимая кнопку **F10** (пошагово), пронаблюдать новое поведение на порте (рис. 33). Светодиод на ножке 0 погас.



Рис. 33 – Имитация светодиодов

Модифицировать инициализацию порта D, поменяв направление на выводе PIND3. Отразить в отчете, какими C и ассемблерными командами происходит считывание данных с порта.

Задание 5. Изучение массивов, семисегментных индикаторов

Модифицировать программу, добавив после команд #include:

```
#include "compiler.h"
#include <avr/interrupt.h>
#include "conf_example.h"
```

Массив из двух переменных:

const char dig[2] = {0b00000111, 0b00111111};

Изменить бесконечный цикл while(1):

```
while (1)
{
     PORTD = dig[0];
     asm("nop");
}
```

Откомпилировать и перейти в режим отладки, понаблюдав, что происходит на порте D (рис. 34).

while (1)	Name	Address	Value	Bits	
	I/O PIND	0x29	0x07		~
PORID = dig[0];	10 DDRD	0x2A	0xFF		
}	VO PORTD	0x2B	0x07		

Рис. 34 – Эффект ёлочки

Открыть окно дизассемблера, найти строчку, отвечающую за вывод в порт значений из массива (рис. 35).

*	MEGA_GPI	O_EXAMPLE3 (Deb	ugging) - AtmelStudio		
File	e Edit Viev	w VAssistX ASF Pr	roject Build Debug Tools Wind	dow	Help
	त्र में आ	- 🕞 💷 📾 🗶		8	💁 🔄 🕅 Debug 🚽 🎮 🗒 🗄 🚑 🚝 📛 💷 💭 🗒
				<u> </u>	
	🔽 🖾 🐝 "		a =[: № • •> 0 • •a	9 <u>=</u>	
Di	isassembly $ imes$	mega_gpio_example.	c conf_example.h	•	IO View - 🕂 🗙
A	ddress: main	ı		•	Filter:
6	Viewing Opt	tions			Name Value
	}			^	AD_CONVERTER
	00000036	POP R24 P	op register from stack		ANALOG_COMPARATOR
	00000037	POP RØ P	op register from stack		🗄 🛄 CPU
	0000038	OUT 0x3F,R0	Out to I/O location		EEPROM
	00000039	POP RØ P	op register from stack		EXTERNAL_INTERRUPT
	000003A	POP R1 P	op register from stack		IVO PORTB
	0000003B	RETI Inter	rupt return		PORTC
	DDRD	= 0xff;			VO PORTD
	0000003C	SER R24 S	et Register		🕀 🗎 SPI
	0000003D	OUT 0x0A,R24	Out to I/O location		
	PORTD) = 0xff;			⊞
	0000003E	OUT 0x0B,R24	Out to I/O location		⊞
	PORTD) = 0;		Ξ	🗉 🗎 TWI
	0000003F	OUT 0x0B,R1	Out to I/O location		🗉 🗎 USARTO
	PORTB	3 = 0xtt;			Name Address Value Bits
	00000040	OU⊤ 0x05,R24	Out to I/O location		
	val =	PINB;			
	00000041	IN R24,0x03	In from 1/0 location		
	PORTD) = dig[0];	Lond Ameridants		
-	00000042	LDI R30,0X00	Load immediate		
	00000043	LDI K31,0X01	Load immediate		×
	NO SO	UPD B24 710	Load indinact with disc		📙 IO View 🔍 ASF Ex 📓 Processor 💐 Solution 🚰 Properti
	00000044	OUT AVAR D24	Out to I/O location	, 	
	00000043	melStudioProgect	my/MEGA GPTO EXAMPLES/ME		
	asm("	'non").	S_IIIY (HEGA_GPIO_EXAMPLES (HE	1	R00 = 0000 R01 = 0000 R02 = 0000 R03 = 000
	00000046	NOP No on	veration		R04 = 0000 R05 = 0000 R00 = 0000 R07 = 0000
	00000047	RJMP_PC-0x0003	Relative jump		$P_{12} = 0.00 P_{13} = 0.00 P_{14} = 0.00 P_{15} = 0.00$
	No so	ource file	Jump		$P_{12} = 0x00 P_{13} = 0x00 R_{14} = 0x00 R_{13} = 0x00$
	00000048	CLI Globa	l Interrupt Disable		$R_{10} = 0x00 R_{11} = 0x01 R_{10} = 0x00 R_{13} = 0x00$ $R_{20} = 0x00 R_{21} = 0x00 R_{22} = 0x00 R_{23} = 0x00$
	00000049	RJMP PC-0x0000	Relative jump		$R_{20}^{2} = 0x00 R_{21}^{2} = 0x00 R_{22}^{2} = 0x00 R_{23}^{2} = 0x00$ $R_{24}^{2} = 0x00 R_{25}^{2} = 0x00 R_{25}^{2} = 0x01$
	0000004A	CPI R16,0xF7	Compare with immediate	-	$R_{24} = 0x65 R_{25} = 0x66 R_{25} = 0x61 R_{25} = 0x61$ $R_{28} = 0xFE R_{29} = 0x04 R_{30} = 0x96$
<		-			R31 = 0x00
W	atch 2		- 4	×	
-	Auton 🔤	Lacala 📈 Watch 1	El Watch 2		
Re	ady				

Рис. 35 – Трансляция кода программы

Задание:

– Добавить в свой вариант изменение направления эффекта при нажатии на кнопку PIN.7(PORTB) и возвращении направления при отпускании.

Дополнительно сформировать на половине порта В бегущую единицу (...0001 -> 0010 -> 0100 -> 1000 -> 0001...), эмулирующую динамический опрос клавиатуры.

Дополнительно на порту С, к которому подключен семисегмент ный индикатор, сформировать индикацию десятичного счета от 0 до 9,
 т. е. выводить соответствующие коды цифр индикатора.

- Привести в отчете код ассемблера итоговой программы.

Контрольные практические задания¹

1. Преобразовать дополнительный код числа одной переменной (байт) в прямой другой переменной.

2. Преобразовать двоичный код в одной переменной (от 0 до 99) в двоично-десятичный другой переменной.

3. Просуммировать два числа в двоичном коде. Сумму, большую 255, заменить байтом единиц.

4. Вычесть два числа в двоичном коде. Разность, меньшую нуля, заменить байтом нулей.

5. Умножить два 8-разрядных числа в двоичном коде. Старший байт заменить байтом единиц.

6. Поделить два 8-разрядных числа в двоичном коде без остатка. Старший байт заменить байтом единиц.

Проверить число А на четность. Если четное, то выставить 4-й бит
 в переменной Б, иначе сбросить в 0.

8. Сложить два десятичных числа (байт) в двоично-десятичном коде.

9. Реализовать суммирующий двоично-десятичный счетчик.

10. Реализовать вычитающий двоично-десятичный счетчик.

¹ Задания служат для контроля усвоения материала студентом. Не являются обязательными для выполнения. Могут быть предложены преподавателем в качестве дополнительного задания при защите лабораторной работы.

Контрольные теоретические вопросы

✓ Какие регистры обслуживают параллельный порт D микроконтроллера ATmega16? Как настроить линию порта на ввод или вывод? Как подключить к линии порта, настроенной на ввод, подтягивающий резистор?

✓ Назовите нагрузочную способность линий порта AVR.

✓ Приведите схему алгоритма разработанной по вашему варианту программы.

 ✓ Запишите результат выполнения арифметических операций: 245/37 и 245%37.

✓ Какими командами можно организовать задержку в одну секунду в программе для AVR на языке Си?

Содержание отчета

Отчет должен содержать листинги отлаживаемых программ на языке С и код ассемблера, комментарии по ходу выполнения пунктов работы, примеры экранных форм (скриншоты), создаваемые по ходу выполнения работы (включая промежуточные действия и финальные задания), отображающие окна регистров и памяти, а также ответы на контрольные вопросы.

2.2 Лабораторная работа № 2 «Изучение прерываний, АЦП, UART»

Цель работы

Целью лабораторной работы является имитация цифрового вольтметра на базе микроконтроллера ATmega16 с отправкой информации на персональный компьютер по интерфейсу RS-232/USB. Имитация подачи аналогового сигнала происходит с помощью регистра данных на линиях порта A (ADC0-ADC7).

Краткая теория

Аналого-цифровой преобразователь (A/D CONVERTER) служит для получения числового значения напряжения, поданного на его вход. Этот результат сохраняется в регистре данных АЦП (рис. 36). Какой из выводов («пинов») микроконтроллера будет являться входом АЦП, определяется числом, занесенным в соответствующий регистр. Величина опорного напряжения (5 В) может изменяться для повышения точности.



Рис. 36 – АЦП-преобразование

Управление АЦП происходит четырьмя регистрами:

– ADMUX (ADC Multiplexer Select Register) – регистр выбора мультиплексора ADC;

– ADCSRA (ADC Control and Status Register) – регистр управления и состояния ADC;

- ADC (ADCL и ADCH) – регистры данных ADC;

- SFIOR – регистр специальных функций ввода-вывода.

Универсальный асинхронный или универсальный синхронно/асинхронный приемопередатчик (Universal Synchronous/Asynchronous Receiver and Transmitter – UART или USART) – удобный и простой последовательный интерфейс для организации информационного канала обмена микроконтроллера с внешним миром.

Он совместим с протоколом стандарта RS-232, что обеспечивает возможность организации связи с персональным компьютером. Для стыковки МК и компьютера обязательно понадобится схема сопряжения уровней сигналов. Для этого существуют специальные микросхемы, например MAX232 (ADM232).

В микроконтроллере для работы с модулем USART используются 6 регистров:

- управляющий регистр UCSRA,

- управляющий регистр UCSRB,

– управляющий регистр UCSRC,

- регистры скорости передачи UBRRL и UBRRH,

– регистр данных UDR.

Система прерываний (INTERRUPTS) – одна из важнейших частей микроконтроллера.

Все микроконтроллеры AVR имеют многоуровневую систему прерываний. Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием. Для каждого такого события разрабатывается отдельная процедура (либо функция), которую называют подпрограммой обработки запроса на прерывание (для краткости – подпрограммой прерывания), и размещается в памяти программ. При возникновении события, вызывающего прерывание, микроконтроллер сохраняет в стеке содержимое счетчика команд, прерывает выполнение центральным процессором текущей программы и переходит к выполнению подпрограммы обработки прерывания. После выполнения подпрограммы прерывания осуществляется восстановление предварительно сохраненного в стеке значения счетчика команд и процессор возвращается к выполнению прерванной программы. Таким образом, нет необходимости непрерывного контроля состояния флагов командами if(), while().

Программа работы

Запустить Atmel Studio 6.2 (не ниже). Выбрать Создать новый пустой проект (рис. 37).

۰	SccApplication1 - AtmelStudio								
File	Edit View VAssistX ASF Pro	ject Build Debug	, т	ools Window Help					
	New	•	ö	Project	Ctrl+Shift+N	🔹 🌁 ISR			
	Open	•	۳ì	File	Ctrl+N				
	Add	•	₫	Example Project	Ctrl+Shift+E				
	Close					-			
đ	Close Solution								
	Import	•							
	Save Selected Items	Ctrl+S							
	Save Output As								
Ø	Save All	Ctrl+Shift+S							
	Export Template								
	Page Setup								
3	Print	Ctrl+P							
	Recent Files	+							
	Recent Projects and Solutions	•							
	Exit	Alt+F4							

Рис. 37 – Создание нового проекта

В открывшемся окно выбрать GCC C Executable Project (Исполняемый проект на языке C). В нижней части указать имя проекта и путь (рис. 38). Они не должны содержать русских символов. Нажать ОК при готовности.

New Project				? 🛛
Recent Templates		Sort by: Default		Search Installed Templates
Installed Templates		GCC C ASF Board Project	C/C++	Type: C/C++ Creates an AVR 8-bit or AVR /ARM 32-bit C
Assembler Atmel Studio Solution	1	GCC C Executable Project	C/C++	project
		GCC C Static Library Project	C/C++	
		GCC C++ Executable Project	C/C++	
		GCC C++ Static Library Project	C/C++	
				einclude cavr/io.hs int main(void) } printf("Hello GCC
Name:	GccApplication2			
Location:	C:\AtmelStudioPro	ogects_my\		Browse
Solution:	Create new solution	on	×	
Solution name:	GccApplication2			Create directory for solution
				OK Cancel

Рис. 38 – Выбор типа проекта

Выбрать в выпадающем меню «megaAVR, 8bit», контроллер ATmega16 (рис. 39).

Device Family:	megaAVR, 8-bit 🗸 🗸					Search for device	۶
Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)		Device Info:		
ATA5833	20	1024	1152	^	Device Name:	ATmega16	
ATA6285	8	512	320	-1			
ATA6286	8	512	320		Speed:	0	
ATA6612C	8	1024	512		Vcc:	2.7/5.5	
ATA6613C	16	1024	512		Eamily:	meraA\/R	
ATA6614Q	32	2048	1024			magantit	
ATmega128	128	4096	4096		Datashee	<u>ts</u>	
ATmega1280	128	8192	4096				
ATmega1281	128	8192	4096		Supported Too	ls	
ATmega1284	128	16384	4096		Atmel-ICE		
ATmega1284P	128	16384	4096		- Interior		
ATmega1284RFR2	128	16384	4096		AVR Drago	<u>on</u>	
ATmega128A	128	4096	4096		AVRISP m	kli	
ATmega128RFA1	128	16384	4096		_		
ATmega128RFR2	128	16384	4096		AVR ONE!		
ATmega16	16	1024	512		JTAGICE3		
ATmega162	16	1024	512			·	
ATmega164A	16	1024	512		JTAGICE I	<u>nkii</u>	
ATmega164P	16	1024	512		The Simulator		
ATmega164PA	16	1024	512		all other		
ATmega165A	16	1024	512		* <u>STK500</u>		
ATmega165P	16	1024	512	~			
<			>				
- <u>1</u>							

Рис. 39 – Выбор микроконтроллера

В открывшийся пустой проект скопировать приведенный текст про-

граммы.

```
#include <avr/io.h>
#include <avr/interrupt.h>
/* макроопределение, для работы с битами */
#define BIT(n)
                 (1 < < (n))
#define ENABLE(x,n) ((x) \models BIT(n))
#define CHECKBIT(x,n) ((x) & BIT(n))
char ADCdata;
                // глобальная переменная для хранения измеренных дан-
ных
void port_init(void)
ł
PORTA = 0x00;
DDRA = 0x00; // порт А делаем входным
PORTB = 0x00; // остальные порты не трогаем
DDRB = 0x00:
PORTC = 0x00;
DDRC = 0x00:
PORTD = 0x00;
DDRD = 0x00:
}
// ADC initialize // Conversion time: 104uS
void adc_init(void)
{ // инициализация АЦП модуля
ADCSRA = 0x00; // disable adc
ADMUX = 0x00|(1 << ADLAR); /* будем использовать первый канал АЦП,
в качестве референса потенциал поданный на вывод AREF, 8-бит разряд-
ность АЦП и равнение налево :) */
ACSR = 0x80; // выключаем аналоговый компаратор
ADCSRA = 0xCD; /* включаем AIIII и запускаем одиночное преобразова-
ние, включаем прерывание по окончанию преобразования, устанавливаем
частоту преобразования */
}
ISR(ADC_vect)
ł
ADCdata = ADCH; // отсылаем данные
ENABLE(ADCSRA, ADSC); // запускаем новое преобразование
```

```
void init_devices(void)
ł
cli(); // на время инициализации периферии запрещаем все прерывания
port_init();
adc_init();
MCUCR = 0x00; // обнуляем на всякий случай все остальные прерывания
GICR = 0x00;
TIMSK = 0x00; // timer interrupt sources
sei(); // разрешаем обратно все прерывания
}
int main(void)
ł
init_devices();
while(1) // создаем бесконечный цикл
{
}
```

Задание 1. Изучение АЦП

Установить брейкпоинт на строке ADCData = ADCH. Откомпилировать проект, нажав **F7 (Built)** и при отсутствии ошибок зайти в режим отладки **F5**.

Для отслеживания созданных переменных существуют окна **Watch1,2**. Обычно они располагаются в левой нижней рабочей области компилятора. При отсутствии, их можно добавить через верхнее выпадающее меню Window.

В окно **Watch** ввести созданную переменную ADCdata, открыть регистры, отвечающие за работу ADC – AD_CONVERTER (рис. 40). Таким образом, в окне **Watch** можно наблюдать и изменять значения ваших переменных, а в окне **I/O View** – регистров микропроцессора.

🗣 GccApplication2 (Debugging) - AtmelStudio 🛛 🕞 🗖						
File Edit View VAssistX ASF Project Build Debug Tools Window						
: ● 코 맨 & 와 이 A 것 A I I M 크 A H M A I M A I M A I M A I M A I M A I M A I M A I M A I M A I M A I M A I M A I M						
: 🔽 🚾 🦓 🖉 🗆 🗆 🗸 🖉 🖓 🦉 🖓 👘						
Disassembly GccApplication2.c ×	IO View 👻 🕂 🗸					
[] { // инициализация АЦП модуля	Filter:					
□ ADMUX = 0x00, // disable add □ ADMUX = 0x00 (1< <adlar); *="" p="" будем="" г<="" использовать=""></adlar);>	Name Value					
в качестве референса потенциал поданный на вывод	AD_CONVERTER					
8-бит разрядность АЦП и равнение на лево :) */	ADC Prescaler Select Bits (ADC 0x05 V					
□ ADCSRA = 0xCD; /* включаем АЦП и запускаем одинс	■ ADC ANALOG COMPARATOR					
включаем прерывание по окончанию преобразования,	BOOT_LOAD					
устанавливаем частоту преобразования */	E CPU					
□ ISR(ADC_vect)						
	VO PORTA					
ADCdata = ADCH; // отсылаем данные	VO PORTB					
	VO PORTC					
100 % - <	PORTD					
Watch 2 🔻 🕂 🗙	Name Address Value Bits					
Name Value Type	ADC 0x24 0x0000					
ADCdata 0 char{data}@0x0060						
	ADLAR 0x01					
	🖪 IO View 🔍 ASF Ex 📓 Processor 💐 Solution 🖀 Properti					
	Registers 🝷 🕂 🗙					
🖼 Autos 👼 Locals 🖉 Watch 1 🛃 Watch 2						
Stopped	at					

Рис. 40 – Считывание значений АЦП

Изменить значение ADC (биты) в правом окне на любое и нажать **F10**. Посмотреть значение переменной в левом нижнем окне.

Перейти в окно дизассемблера. Найти команды чтения данных из регистра ADCH, как компилятор считал данные только из 1-го регистра (ADCH) без чтения ADCL (рис. 41). Найти в дизассемблере и указать в отчете как работает макрос ENABLE(ADCSRA, ADSC).



Рис. 41 – Окно дизассемблера копирования данных из АЦП

Выполнить задание согласно варианту:

1. Настроить АЦП на автоматический циклический запуск (без команды ENABLE(ADCSRA, ADSC);).

2. Настроить коэффициент усиления 10х.

3. Изменить вход АЦП на ADC3.

4. Изменить канал опорного напряжения на AVCC с внешним конденсатором на выводе AREF.

5. Изменить канал опорного напряжения на внутренний источник опорного напряжения 2.56 В с внешним конденсатором на выводе AREF.

6. Уменьшить скорость работы в 64 раза.

7. Уменьшить скорость работы в 32 раза.

8. Уменьшить скорость работы в 16 раз.

9. Установить дифференциальный вход.

10. Изменить порядок считанных данных на противоположный (результат преобразования может иметь левосторонний или правосторонний формат).

Задание 2. Изучение UART

Добавим передачу оцифрованных данных по UART (предположив, что подключен компьютер через RS-232 преобразователь).

Добавить функцию (т. к. в скобке справа usart_init указана переменная) инициализации UART в любое место между char ADCdata и процедурой main().

// USART initialize
// desired baud rate: 9600 actual: baud rate: 9615 (0,2%)
// char size: 8 bit parity: Disabled
void usart_init(unsigned baudrate)
{
 UCSRB = 0x00; // disable while setting baud rate
 UCSRA = 0x00;
 UCSRC = BIT(URSEL) | 0x06;

Модифицировать код в прерывании.

ISR(ADC_vect)

while(!CHECKBIT(UCSRA,UDRE)); // ждем, пока освободится буфер UDR = ADCH; // отсылаем данные ENABLE(ADCSRA, ADSC); // запускаем новое преобразование

Добавить вызов инициализации.

Переменная baudrate в функции usart_init заносится в регистр UBRRL, таким образом, число 25 в данном примере соответствует настройкам скорости. Посмотреть описание регистра UBRRL.

usart_init(25);

Установить «брейкпоинт» на строке UDR = ADCH. Откомпилировать проект, при отсутствии ошибок перейти в режим отладки (F5) (рис. 42). Имитируя вводом в регистр ADC оцифрованные данные, посмотреть результат в режиме отладки.

¢	□ ISR(ADC_vec { while(!CH UDR = ADCH ENABLE(ADC }	t) ECKBIT(UCSRA,UDRE) <mark>;// отсылаем данны</mark> SRA, ADSC);// запу); // ждем пока осы ме /скаем новое преобра У
1	100 % - <		>
W	Vatch 2		- ₽ ×
	Name	Value	Туре 🔥
	ADCdata	12	char{prog}@0x0000

Рис. 42 – Вывод данных в параллельный порт

Какие команды ассемблера используются для передачи между регистрами (рис. 43)?

0)isassembly ×	GccApplication2.c	<u>د</u> .	•
	Address:ve	ector_14	-	•
C	Viewing Opt	ions		
	0000005B	CLR R1	Clear Register	^
	0000005C	PUSH R24	Push register on stack	
	while(!	CHECKBIT(UCSR/	A,UDRE)); // 2222 2222 2222	
	0000005D	SBIS 0x0B,5	Skip if bit in I/O regi	
	0000005E	RJMP PC-0x00	01 Relative jump	
	UDR = AD	CH; // 00000		Ξ
¢	0000005F	IN R24,0x05	In from I/O location	
	00000060	OUT 0x0C,R24	Out to I/O location	
	ENABLE (A	DCSRA, ADSC);	// 22222222 22222 2222222	
	00000061	SBI 0x06,6	Set bit in I/O register	
	}			
	00000062	POP R24	Pop register from stack	
	00000063	POP RØ	Pop register from stack	~
<	:	11.1		

Рис. 43 – Отправка измеренных данных АЦП в порт UART

Найти и пояснить, как выполняется команда (рис. 44). UCSRC = BIT(URSEL) | 0x06

Di	sassembly ×	GccApplication2.c	•
A	ddress: usa	t_init	•
6	Viewing Op	tions	
	UCSRB =	0x00; //disable while setting baud rate	^
	00000047	OUT 0x0A,R1 Out to I/O location	_
	UCSRA =	0x00;	
	00000048	OUT 0x0B,R1 Out to I/O location	
	UCSRC =	BIT(URSEL) 0x06;	
	00000049	Load immediate	-
	0000004A	Out to I/O location	
	UBRRL =	baudrate;	

Рис. 44 – Изучение использования маски

Задание

– Изменить в своем проекте скорость передачи данных по UART в 2 раза быстрее.

Дополнительно сделать усреднение результата оцифровки (среднее арифметическое) по четырем измерениям. Передавать усредненные данные после четырех измерений.

– Дополнительно добавить индикацию светодиодом по окончании оцифровки. Светодиод должен гореть, пока АЦП остановлено.

– Дополнительно сделать старт АЦП-преобразования и передачу данных после нажатия кнопки.

Контрольные теоретические вопросы

✓ Какой метод аналого-цифрового преобразования сигнала применяется в микроконтроллерах AVR? Назовите другие типы АЦП.

✓ Укажите диапазон тактовой частоты, рекомендуемый для работы АЦП АТmega16.

✓ Назовите возможные режимы работы АЦП.

✓ Какими ассемблерными вставками можно разрешать и запрещать глобально прерывания в программе для AVR на языке Си?

✓ Прокомментировать результат выполнения команды ADCSRA|=0x40;

Содержание отчета

Отчет должен содержать листинги отлаживаемых программ на языке С и код ассемблера, комментарии по ходу выполнения пунктов работы, примеры экранных форм (скриншоты), создаваемые по ходу выполнения работы (включая промежуточные действия и финальные задания), отображающие окна регистров и памяти, а также ответы на контрольные вопросы.

2.3 Лабораторная работа № 3 «Таймеры/счетчики, ШИМ (РWМ) модуляция»

Цель работы

Целью лабораторной работы является исследование работы таймеров/счетчиков и системы прерываний.

Для выполнения работы необходимо использовать «Timer/Counter0/1/2» микроконтроллера. Представить, что выходы «OC1A» и/или «OC1B» связаны со светодиодом/светодиодами. Принять для простоты, что светимость светодиода линейно зависит от коэффициента заполнения прямоугольных импульсов, подаваемых на него.

Краткая теория

Микроконтроллеры AVR имеют в своем составе от 1 до 4 таймеров/счетчиков (TIMER/COUNTERS) с разрядностью 8 или 16 бит, которые могут работать и как таймеры от внутреннего источника тактовой частоты, и как счетчики внешних событий.

Их можно использовать для точного формирования временных интервалов, подсчета импульсов на выводах микроконтроллера, формирования последовательности импульсов. В режиме ШИМ (PWM) таймер/счетчик может представлять собой широтно-импульсный модулятор, он используется для генерирования сигнала с программируемыми частотой и скважностью.

Таймеры/счетчики способны вырабатывать запросы прерываний, переключая процессор на их обслуживание по событиям и освобождая его от необходимости периодического опроса состояния таймеров. Поскольку основное применение микроконтроллеры находят в системах реального времени, таймеры/счетчики являются одним из наиболее важных элементов. Таймер-счетчик Т0 использует два вывода микроконтроллера АТтеga16. Вывод Т0 (PB0) – это вход внешнего тактового сигнала. Он может применяться, например, для подсчета импульсов. Вывод ОС0 (PB3) – это выход схемы сравнения таймера-счетчика. На этом выводе с помощью таймера он может формировать меандр или ШИМ-сигнал. Также он может просто менять свое состояние при срабатывании схемы сравнения (рис. 45).



Рис. 45 – Выводы микросхемы ATmega16

Управление работой таймера/счетчика 1 осуществляется с помощью регистров, описанных в таблице 1. Настройка таймеров 0/2 происходит в своих регистрах.

Таблица 1 – Описание регистров управления таймера/счетчика1

Регистр	Краткое описание
TCNT1	Timer/Counter1 – регистр, содержащий текущее значение тай-
	мера/ счетчика 1
TCCR1A	Timer/Counter1 Control Register А – регистр задания режимов
	таймера/счетчика 1
TCCR1B	Timer/Counter1 Control Register В – регистр задания режимов
	таймера/счетчика 1

Регистр	Краткое описание
	Timer/Counter Output1 Compare Register А – выходной регистр
OCR1B	компаратора А
	Timer/Counter Output1 Compare Register В – выходной регистр
OCKID	компаратора В
ICD 1	Timer/Counter1 Input Capture Register1 – входной регистр за-
ICKI	щелки 1-го таймера
TIMSK	Timer Interrupt Mask Register – регистр маски прерываний
TIFR	Timer Interrupt Flag Register – регистр флагов прерываний

Окончание табл. 1

Программа работы

Запустить Atmel Studio. Выбрать **File – New – Project...** – новый пустой проект (рис. 46).

۰	ccApplication1 - AtmelStudio)				
File	Edit View VAssistX ASF Proj	ject Build Debug	То	ools Window Help		
	New	•	٦Ĵ	Project	Ctrl+Shift+N	🔹 🏄 ISR
	Open	•	ð	File	Ctrl+N	. •
	Add	► c	₽	Example Project	Ctrl+Shift+E	
	Close					
67	Close Solution					
	Import	•				
	Save Selected Items	Ctrl+S				
	Save Output As					
1	Save All	Ctrl+Shift+S				
	Export Template					
	Page Setup					
3	Print	Ctrl+P				
	Recent Files	•				
	Recent Projects and Solutions	•				
	Exit	Alt+F4				

Рис. 46 – Окно создания проекта

В открывшемся окно выбрать **GCC C Executable Project** (Исполняемый проект на языке C). В нижней части указать имя проекта и путь (рис. 47). Они не должны содержать русских символов. Нажать **ОК** при готовности.

New Project						? 🔀
Recent Templates		Sort by: De	fault 💽 🛄		Search Installed Templates	م
Installed Templates C/C++		GC GC	CC C ASF Board Project	C/C++	Type: C/C++ Creates an AVR 8-bit or AVR	ARM 32-bit C
Assembler Atmel Studio Solutior	ı	GC	CC C Executable Project	C/C++	project	
		GC	CC C Static Library Project	C/C++		
		GCC GC	CC C++ Executable Project	C/C++		
		ecc GC	CC C++ Static Library Project	C/C++		
					sinclude cavr, int main(void) { printf("Hello	(io. hs
Name:	GccApplication2					
Location:	C:\AtmelStudioPro	ogects_my\			Browse	
Solution:	Create new solution	on		~	Consta disertary for colution	
Solution name:	GCCApplication2				Create directory for solution	
					ОК	Cancel

Рис. 47 – Выбор типа проекта

В следующем окне в выпадающем меню выбрать «megaAVR, 8bit», контроллер ATmega16.

В открывшийся пустой проект скопировать приведенный текст программы.

#include <avr/io.h>
#include <avr/interrupt.h>
void init_timer1(void) // Инициализация таймера/счетчика21
{
 DDRB = (1 << PB0); // настраиваем PB0 на выход
 TCCR1B = (0<<CS12)|(0<<CS11)|(1<<CS10); // настраиваем делитель
 TIMSK |= (1<<TOIE1); // разрешаем прерывание по переполнению
таймера
 TCNT1 = 64456; // выставляем начальное значение TCNT1
}
ISR(TIMER2_OVF_vect)
{
 if ((PORTB & 3) == 1)</pre>

```
{
  PORTB &= (0xFF << 2); // Отключение выводов PB0, PB1
                      // Включение РВ1
  PORTB |= 2;
 }
 else
 {
  PORTB &= (0xFF << 2); // Отключение выводов PB0, PB1
  PORTB |= 1;
                // Включение РВО
 }
}
ISR( TIMER1_OVF_vect )
{
  TCNT1 = 64456; // выставляем начальное значение TCNT1
  if( PINB & ( 1 << PB0 ) ) {
    PORTB &= ~(1 << PB0);
  }
  else {
    PORTB |= ( 1 << PB0 );
  }
}
int main()
ł
  init_timer1();
               // выставляем бит общего разрешения прерываний
  sei();
  while(1)
  ł
    asm("nop");
           // вечный цикл
  };
return 0;
```

Задание 1. Изучение прерывания по переполнению

Установить «брейкпоинт» на строке ISR (рис. 48). Откомпилировать проект, нажав **F7 (Built)**, и при отсутствии ошибок зайти в режим отладки **F5**.

GccApplication3 (Debugging) - AtmelStudio	
File Edit View VAssistX ASF Project Build Debug Tools Window Help	
: M - A M - 🐸 🖬 🖉 🖉 🗛 🖄 - M - A - A - A - A - A - A - A - A - A	MI Debug - DM TIMERO OVE vect -
ji 🔽 🗁 🖓 🍋 🖁 🧐 🤐 🄐 📑 🗐 💷 🔷 🖉 🖓 🖼 🚰 🗍	🖅 T Hex 🐚 T 🚽 🖏 🖾 💷 🥋 🗳 🚽 🗮 🚵 🚽 🖓
Disassembly GccApplication3.c ×	IO View
In the second se	Filter:
÷	Name Value
ISR(TIMER1_OVF_vect)	🖹 Waveform Generation Mo 🛛 0x00 🔽
	Clock Selects (TCCR0)
TCNT1 = 64456; //выставляем начальное значение TCNT1	O TIMER_COUNTER_1
if(PINB & (1 << PB0)) {	Prescaler source of Timer 0x01 🔽
PORTB &= ~(1 << PB0);	O TIMER_COUNTER_2
) else /	Name Address Value Bits
PORTB = (1 << PB0):	Address Value Dits
}	
}	
⊡int main()	
{{	
100 % - <	☐ ICES1 0x00 □ □ □ □ □ □ □ □ □
wate 2 T	🖹 WGM1 0x00
Watch 2	Ocs1 0x01
value Type	
	⊞ 🕑 TIFR 0x58 0x18 🛛 🗖 🗖 🗖 🖉
	⊕ TIMSK 0x59 0x04 □ □ □ □ ■ ■ □
	🔄 IO View 🔍 ASF Ex 🐺 Processor 🛛 Solution 🖀 Properti
	Memory 2
🖼 Autos 👼 Locals 👰 Watch 1 🛃 Watch 2	
Ready	

Рис. 48 – Настройка таймера/счетчика 1

Посмотреть состояние регистров (рис. 49). В каком режиме работает таймер, до какого значения происходит счет? Что происходит с выводом PB0?

٢	⊟ ISR	(TIMER1_OVF_vect) TCNT1 = 64456; //выставляем начальное значение TCNT1 if(PINB & (1 << PB0)) { PORTB &= ~(1 << PB0);	VO PORTB VO PORTC VO PORTD I I SPI				~
		}	Name	Address	Value	Bits	
		else {	1/O PINB	0x36	0x00	0000000	^
		PORTB = (1 << PB0);	1/O DDRB	0x37	0x01		
	}	}	VO PORTB	0x38	0x00	00000000	

Рис. 49 – Состояние порта В

Установить «брейкпоинт» на пустой команде asm («пор»). Нажимая **F10** (пошагово), посмотреть, как происходит счет в регистре TCNT1 (рис. 50). Когда происходит выставление флагов таймера 1?



Рис. 50 – Работа таймера/счетчика

Задание 2. Изучение двух работающих таймеров

Добавить процедуры инициализации таймера 0, 1, 2 перед процедурой main().

```
void init timer0(void) // Инициализация таймера/счетчика0
 {
  OCR0 = 255:
                    // Содержимое регистра сравнения
  // Задаем режим работы таймера
  TCCR0 = (0 \iff WGM01) | (0 \iff COM00) | (0 \iff CS02) | (1 \iff CS00);
 }
 void init_timer1(void) // Инициализация таймера/счетчика1
 ł
  DDRB = (1 \ll PB0); // настраиваем PB0 на выход
  TCCR1B = (0 << CS12) | (0 << CS11) | (1 << CS10); // настраиваем делитель
  TIMSK |= (1 << TOIE1); // разрешаем прерывание по переполнению тай-
мера
  TCNT1 = 64456;
                      // выставляем начальное значение TCNT1
 }
 void init timer2(void) // Инициализация таймера/счетчика2
 {
  OCR2 = 255;
  TCCR2 = (0 \iff WGM21) | (0 \iff CS22) | (0 \iff CS21) | (1 \iff CS20);
 // TIMSK |= (1 << OCIE2); // Устанавливаем для него прерывание совпа-
дения
 }
 ISR(TIMER2 OVF vect)
 ł
  if ((PORTB \& 3) == 1)
   PORTB &= (0xFF << 2); // Отключение выводов PB0, PB1
   PORTB |= 2;
                // Включение РВ1
```

```
else
{
    PORTB &= (0xFF << 2); // Отключение выводов PB0, PB1
    PORTB |= 1; // Включение PB0
  }
```

В основную программу добавить вызов процедур инициализации таймеров 0,2.

```
init_timer2();
init_timer0();
```

Установить брейкпоинт на пустой команде asm («nop»). Нажимая кнопку **F10** (пошагово), посмотреть, что происходит с таймерами 0, 1, 2 в регистрах TCNT0, 1, 2. Какое вызывается прерывание?

Добавить в проект второе прерывание по таймеру 0.

```
void init_timer0(void) // Инициализация таймера/счетчика 0
{
     OCR0 = 255; // Содержимое регистра сравнения
     // Задаем режим работы таймера
     TCCR0 = (1 << WGM01) | (0 << COM00) | (0 << CS02) | (1 << CS00);
     TIMSK |= (1 << TOIE0); // Устанавливаем для него прерывание по
     nереполнению
   }
   ISR(TIMER0_OVF_vect)
   {
        asm("nop");
   }
}</pre>
```

Зайти в режим отладки, посмотреть, как срабатывают два прерывания. В каких режимах работают таймеры? При каких условиях? До какого значения происходит счет? Какие флаги выставляются?

Задание 3. Изучение режима сравнения – ШИМ

Модифицировать процедуру инициализации таймера 0 и, добавив дополнительное прерывание по сравнению, установить OCR0 = 127.

```
void init_timer0(void) // Инициализация таймера/счетчика 0
{
    OCR0 = 127;//255; // Содержимое регистра сравнения
    // Задаем режим работы таймера
    TCCR0 = (1 << WGM01) | (0 << COM00) | (0 << CS02) | (1 << CS00);
    //TIMSK |= (1 << TOIE0); // Устанавливаем для него прерывание по пере-
полнению
    TIMSK |= (1 << OCIE0); // Устанавливаем для него прерывание по срав-
нению
    ISR(TIMER0_OVF_vect)
    {
        asm("nop");
    }
    ISR(TIMER0_COMP_vect)
    {
        asm("nop");
    }
    }
</pre>
```

Зайти в режим отладки (**F5**). Развернуть регистры состояния таймера 0. В пошаговом режиме посмотреть, что происходит с флагом OCF0 (рис. 51).

AVR studio 6.2 имеет программные ошибки при работе с таймерами и прерываниями, которые, возможно, были исправлены в новой версии. При пошаговом режиме флаг сравнения выставляется ПРАВИЛЬНО, однако переход на вектор прерывания не происходит. При нажатии на **F5** (запуск) возможно зависание программы. Выход из режима отладки и вход в него снова помогут избежать зависаний. На официальном сайте разработчика эмулятора указано, что данная ошибка известна.



Рис. 51 – Настройка таймера 0 в режиме сравнения

Модифицировать вектор прерывания таймера 0.

```
ISR(TIMER0_COMP_vect)
{
     OCR0 = OCR0 + 1;
     asm("nop");
}
```

Зайти в режим отладки. В каком режиме работает таймер? При каких условиях? До какого значения происходит счет? Какие флаги выставляются? Посмотреть, когда выставляется флаг OCF0. Данная запись позволяет изменять уровень сравнения для формирования модулированного ШИМ-сигнала. Например, в данной записи происходит постоянное прибавление единицы, что приводит к формированию «пилообразной» развертки. Если требуется сформировать модулированный синусоидальной разверткой ШИМ-сигнал, нужно OCR0 изменять по синусоидальному закону sin (2 × 3.14 × F).

Задание 4. Изучение счетчика импульсов

Изменить настройки таймера 1 в проекте.

```
void init_timer0(void) // Инициализация таймера/счетчика 0
{
    OCR0 = 127;//255; // Содержимое регистра сравнения
    DDRB = 1;
    // Задаем режим работы таймера
    TCCR0 = (1 << WGM01) | (1 << COM00) | (1 << CS02) | (1 << CS01) | (1
<< CS00);
    TIMSK |= (1 << TOIE0); // Устанавливаем для него прерывание по пере-
полнению
    // TIMSK |= (1 << OCIE0); // Устанавливаем для него прерывание по
    cpaвнению
    }
    ISR(TIMER0_OVF_vect)
    {
        asm("nop");
    }
    }
}
</pre>
```

Установить брейкпоинт на пустой команде asm («пор»). Нажимая кнопку **F10** (пошагово), посмотреть, что происходит с таймерам 0 (TCNT0). Какое вызывается прерывание? Посмотреть настройки таймера 0 (рис. 52).



Рис. 52 – Регистры таймера 0

В режиме отладки перейти на регистр порта В. Проимитировать поступление импульса на ножку микроконтроллера. Для этого установить PINB.0 в единичное состояние (рис. 53). Нажать на **F10** (пошагово).

10 PORT	4			
PORTE	3			
VO PORTO	2			
VO PORTE)			
Name	Address	Value	Bits	
1/O PINB	0x36	0x00	00000000	~
1/O DDRB	0x37	0x01		

Рис. 53 – Имитация нажатий кнопки

Перейти на регистр состояния таймера 0 (рис. 54). Как изменилось значение TCNT0?



Рис. 54 – Работа таймера в режиме счета

Сбросить бит порта В. Посмотреть, как изменится значение ТСМТО.

Выполнить задание согласно варианту:

1. На выводе ОСО (PB3) имитировать изменение яркости светодиода (... – плавное нарастание – плавный спад – ...), «новогодняя гирлянда», для одного цвета. Для этого сформировать на ножке ШИМ-сигнал FAST PWM, модулированного треугольной разверткой.

2. На выводе ОС0 (PB3) имитировать изменение яркости светодиода (... – плавное нарастание – плавный спад – ...), «новогодняя гирлянда», для одного цвета. Для этого сформировать на ножке ШИМ-сигнал PWM Phase correct, модулированного синусоидальной разверткой.

3. На выводе ОСО (PB3) имитировать изменение яркости светодиода (... –плавное нарастание – мигнуть 3 раза – плавное нарастание – ...), «новогодняя гирлянда», для одного цвета. Для этого сформировать на ножке ШИМ-сигнал FAST PWM, модулированного нарастающей пилообразной разверткой.

4. На выводе ОСО (PB3) имитировать изменение яркости светодиода (... – лавное гашение – мигнуть 3 раза – плавное гашение – ...), «новогодняя гирлянда», для одного цвета. Для этого сформировать на ножке ШИМ-сигнал FAST PWM, модулированного спадающей пилообразной разверткой.

5. Реализовать подсчет числа импульсов (оборотов двигателя/компьютерного вентилятора), поступающих на ножку процессора с помощью Timer/Counter0 и Timer/Counter1 за 1 секунду. Один таймер ведет счет импульсов, другой отсчитывает 1 секунду.

6. Реализовать измерение длительности импульса (режим захвата таймера), поданного на ножку контроллера с помощью Timer/Counter0 и Timer/Counter1. Один таймер ведет фиксацию интервала, другой ведет измерение длительности.

7. На выводе ОС0 (PB3) имитировать изменение яркости светодиода от числа нажатий на кнопку (управление режимами светодиодного фонарика). Одно нажатие – минимальная яркость, второе нажатие для средней яркости, третье нажатие – максимальная яркость, после четвертого нажатия происходит повтор. Для этого вывести ШИМ-сигнал на вывод ОС0 и изменять скважность от числа нажатий. Максимальное значение ШИМ соответствует максимальной яркости (1), минимальное – минимальной яркости (0), среднее соответствует половинной скважности/яркости (γ = 0.5).

61

8. На выводе ОС0 (PB3) имитировать изменение яркости светодиода от величины измеренных с помощью АЦП-данных (управление яркостью светодиодного фонарика). Для этого запустить АЦП в любом режиме. Максимальное значение АЦП соответствует максимальной яркости (1), минимальное – минимальной яркости (0), среднее соответствует половинной скважности/яркости (γ = 0.5).

9. Сформировать измерение данных АЦП в строго заданные таймером промежутки времени для стандартной частоты записи CDдисков/аудио файлов 44.1 кГц (микрофонная запись голоса CD-диска). Для этого запустить таймер с частотой 44.1 кГц, в котором производить считывание данных с АЦП и запуск АЦП с максимальной скоростью так, чтобы к следующему входу в прерывание по таймеру АЦП закончило преобразование.

10. Сформировать измерение данных АЦП в строго заданные таймером промежутки времени для стандартной частоты записи DVD-аудиодисков/аудиофайлов 96 кГц (микрофонная запись голоса DVD-диска). Для этого запустить таймер с частотой 44.1 кГц, в котором производить считывание данных с АЦП и запуск АЦП с максимальной скоростью так, чтобы к следующему входу в прерывание по таймеру АЦП закончило преобразование.

Контрольные теоретические вопросы

✓ Чем отличаются прерывания INT0 по фронту и по уровню? Какие регистры обслуживают внешние аппаратные прерывания?

✓ Что произойдет, если к кнопке INT0 не подключить резистор подтяжки?

✓ Укажите физические адреса векторов прерывания микроконтроллера ATmega16.

✓ Перечислите виды ШИМ 16-разрядного таймера/счетчика Т/С1 микроконтроллера АТmega16.

Содержание отчета

Отчет должен содержать листинги отлаживаемых программ на языке С и код ассемблера, комментарии по ходу выполнения пунктов работы, примеры экранных форм (скриншоты), создаваемые по ходу выполнения работы (включая промежуточные действия и финальные задания), отображающие окна регистров и памяти, а также ответы на контрольные вопросы.

ЛИТЕРАТУРА

Основная литература

1. Бородин К. В. Микропроцессорные устройства и системы : учеб. пособие / К. В. Бородин. – Томск : ФДО, ТУСУР, 2016. – 137 с.

2. Русанов В. В. Микропроцессорные устройства и системы : учеб. пособие для вузов / В. В. Русанов, М. Ю. Шевелев. – Томск : Томск. ун-т систем упр. и радиоэлектроники, 2006. – 200 с. : ил.

3. Микропроцессорные системы : учеб. пособие для вузов / Е. К. Александров и др. ; под общ. ред. Д. В. Пузанкова. – СПб. : Политехника, 2002. – 935 с. : ил.

4. Шарапов А. В. Микропроцессорные устройства и системы : метод. указания к выполнению курсового проектирования / А. В. Шарапов. – Томск : Томск. ун-т систем упр. и радиоэлектроники, 2008. – 24 с.

5. Рождественский Д. А. Микропроцессорные устройства в системах управления : учеб. пособие / Д. А. Рождественский. – Томск : Том. межвуз. центр дистанционного образования, 2003. – 130 с.

6. Белов А. В. Конструирование устройств на микроконтроллерах /
А. В. Белов. – СПб. : Наука и Техника, 2005. – 256 с. : ил.

Дополнительная литература

7. ATMega16. Datasheet. Atmel CO LTD. – USA, 2001.

8. Официальный сайт компании Atmel. – Режим доступа : http://www.atmel.com (дата обращения: 15.02.2016).